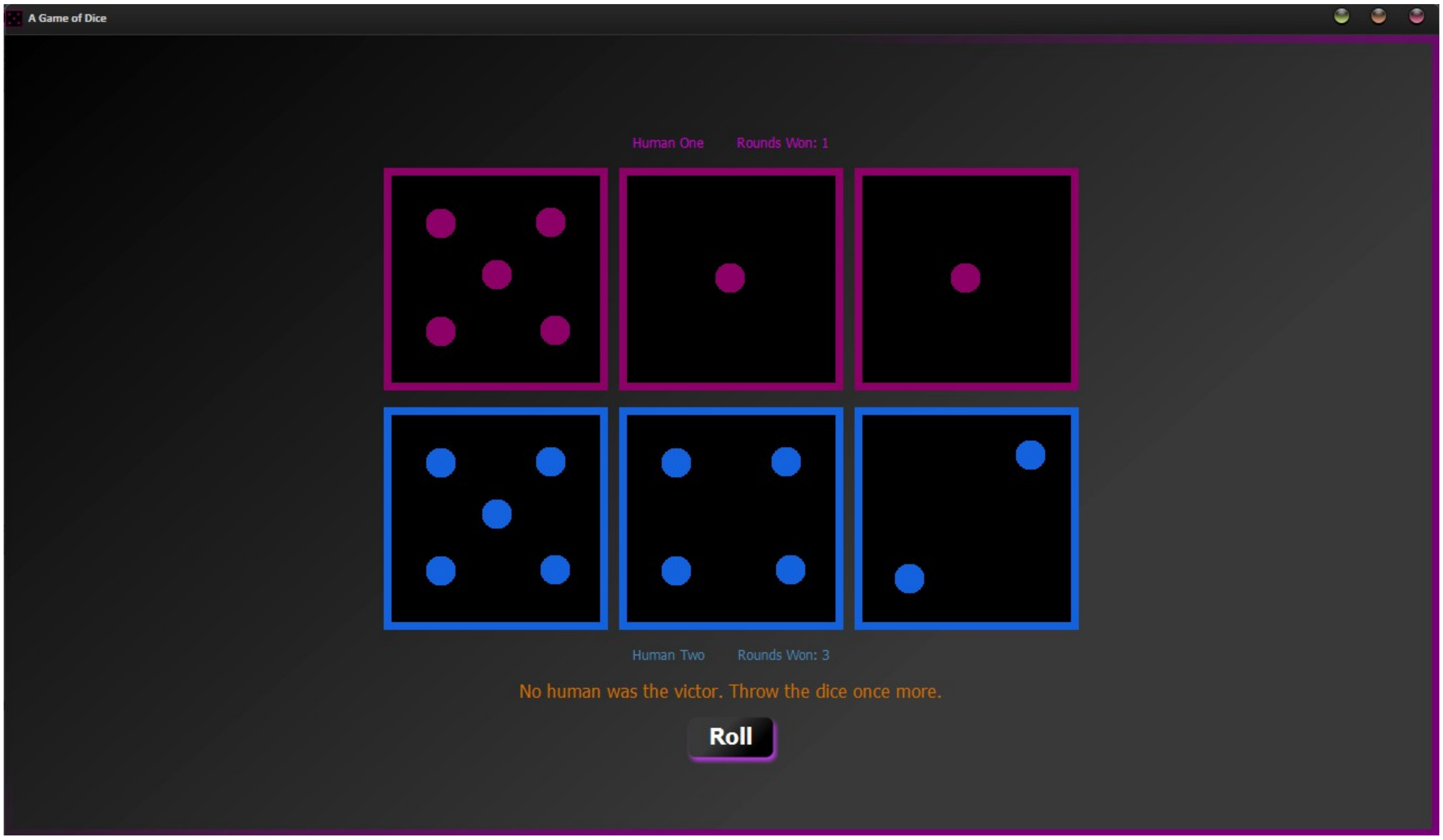
Cee-lo: A Game of Dice

CS242

Project #2

10/31/13



No human was the victor. Throw the dice once more.

by

Corey Richardson

and

Adam Kimball

Design Summary

Cee-lo is actually a pretty simple game, and as such, our design isn't too terribly hard to describe. The game relies on the players rolling dice three dice each, over and over again, to see who wins. Our version of Cee-lo is considered “bankless”, as there are only two participants. There are four possible outcomes(atop a meaningless roll, which a player must roll again), ranked from lowest to highest priority:

1-2-3: If a player rolls a 1, 2, and 3, they automatically lose unless the other does as well.

Point: If a player rolls two of a kind, the third die becomes a “point”. If the other player rolls a point higher than them, rolls **Trips**, or a **4-5-6**, they lose.

Trips: If a player rolls three of a kind, it is considered **Trips**. This can only lose to a **4-5-6**, or another set of **Trips** of higher value.

4-5-6: This is exactly what it sounds like. A player who rolls a 4, a 5, and a 6, automatically wins unless the other player does the same.

For our version, we have only one player roll: the other simply stands by and watches their outcome. Both die have equal probabilities, so the only real difference is that one player rolls for both. Rounds won are displayed to the player, and results can be seen both in the GUI and in a log file that's appended to with every click of the Roll button.

Implementation

Our implementation of Ceelo is pretty simple, albeit branching across quite a few classes. The main class is ***CeeloProject***. It simply sets up the stage for JavaFX, sets the tilebar up, and kicks off the FXML loader. Our primary logic classes are ***Ceelo***, ***Logger***, ***Player***, and ***RollType***, which are all pretty self-explanatory. ***Ceelo*** starts and controls the actual game of Cee-lo that's happening behind the GUI, and is generally the interfacing class for ***Player***. ***RollType*** is simply an enumerator that's referenced repeatedly through our code for the result of a game, so integers don't have to be tossed around in a confusing fashion. ***Logger*** is reported to by several other classes, it's the class that writes everything to an output file. ***Player*** is exactly what it sounds like it would be – it's our representation of a player through code. It's also a private class! Since the code may be a bit strange to look at if you don't have context, I should clarify – the Player class has a private integer of *"extra_value"*, which is a value that's reported with Point and Trips to check which player wins in case of equivalent RollType.

Where things start to get a bit different is with our GUI. We use a combination of FXML and CSS for GUI manipulation, and the code comes out a lot cleaner for it. To that end, we have ***Base.fxml***, ***BaseController.java***, ***DiceBox.fxml***, ***DiceBox.java***, ***dicebox.css,*** and ***styles.css***. There's also a huge slew of PNG files and GIFs, but the file names for them are so self-explanatory that I'm not going to list them here, save to clarify on the rolling GIFs: each player has three roll animations of varying order and speed so the dice rolls don't look synchronized, which is why there are six of them instead of two.

Base and ***BaseController*** are the first pairing. ***Base*** is an FXML file that is essentially our entire GUI: it initializes to a 720p resolution, and has 6 rows with 1 column each. It also contains two ***DiceBox***es, which is how we're able to achieve the effect of appearing to have three dice in one column. ***BaseController*** handles the changing of most major GUI elements, changing the dice around and altering text as needed. ***DiceBox*** acts almost like an array of dice for the screen, and is what is used inside of ***Base*** so that the dice are actually capable of being manipulated. ***DiceBox.java*** is the controller class for it, which is referenced by ***BaseController*** to control the animations in a fashion that doesn't become incredibly convoluted, via its *"setImage"* function, which is the crux of our entire GUI system. ***DiceBox.css*** is actually an empty cascading style sheet, but without it JavaFX wouldn't allow us to use ***DiceBox***. ***Styles.css*** is the flavor style sheet, it dictates the background of the GUI pane, the border it uses, the style/color/size of all three text elements on screen, and everything that's done visually with the button.

Gameplay

The gameplay follows exactly what was specified from a bankless game of Cee-lo on page 1, however, the actual GUI wasn't explained there. The game starts with both win counters at 0, a field of six blank dice, a text prompt reading “Human One: Press the button to roll the dice and engage Ceelo!”, and a roll button. Upon clicking the roll button, the dice will animate: all six will roll, and end with a still image showing what the top value of that die is. At this point the status text will change, to tell the players what the result of the round was: either no clear winner, player 1 winning, or player 2 winning. If one player's dice are not a meaningless roll but another's are, the player with the meaningful roll has their dice 'locked'. Their dice will not roll or animate upon the next click of the roll button, but the other player's will. That player will continue to roll until they get something meaningful. The game does not have any predefined ending, but rather the round counter simply ticks up further and further until one chooses to stop playing. A typical log file is shown on the next page.

```
----- Initializing Ceelo -----

P1 Rolls: A D6 with a top of 4, A D6 with a top of 6, A D6 with a top of 5,
Result: extra_value=0, type=HighSeq

P2 Rolls: A D6 with a top of 6, A D6 with a top of 2, A D6 with a top of 5,
Result: extra_value=0, type=None

No human was the victor. Throw the dice once more.

------------------------------

P2 Rolls: A D6 with a top of 2, A D6 with a top of 2, A D6 with a top of 2,
Result: extra_value=2, type=Trips

Human one has won the round. Perhaps human two should work the button next time?

------------------------------

P1 Rolls: A D6 with a top of 1, A D6 with a top of 3, A D6 with a top of 1,
Result: extra_value=3, type=Point

P2 Rolls: A D6 with a top of 5, A D6 with a top of 5, A D6 with a top of 2,
Result: extra_value=2, type=Point

Human one has won the round. Perhaps human two should work the button next time?

------------------------------

P1 Rolls: A D6 with a top of 4, A D6 with a top of 5, A D6 with a top of 5,
Result: extra_value=4, type=Point

P2 Rolls: A D6 with a top of 1, A D6 with a top of 5, A D6 with a top of 3,
Result: extra_value=2, type=None

No human was the victor. Throw the dice once more.

------------------------------

P2 Rolls: A D6 with a top of 2, A D6 with a top of 4, A D6 with a top of 6,
Result: extra_value=2, type=None

No human was the victor. Throw the dice once more.

------------------------------

P2 Rolls: A D6 with a top of 2, A D6 with a top of 1, A D6 with a top of 2,
Result: extra_value=1, type=Point

Human one has won the round. Perhaps human two should work the button next time?

------------------------------
```